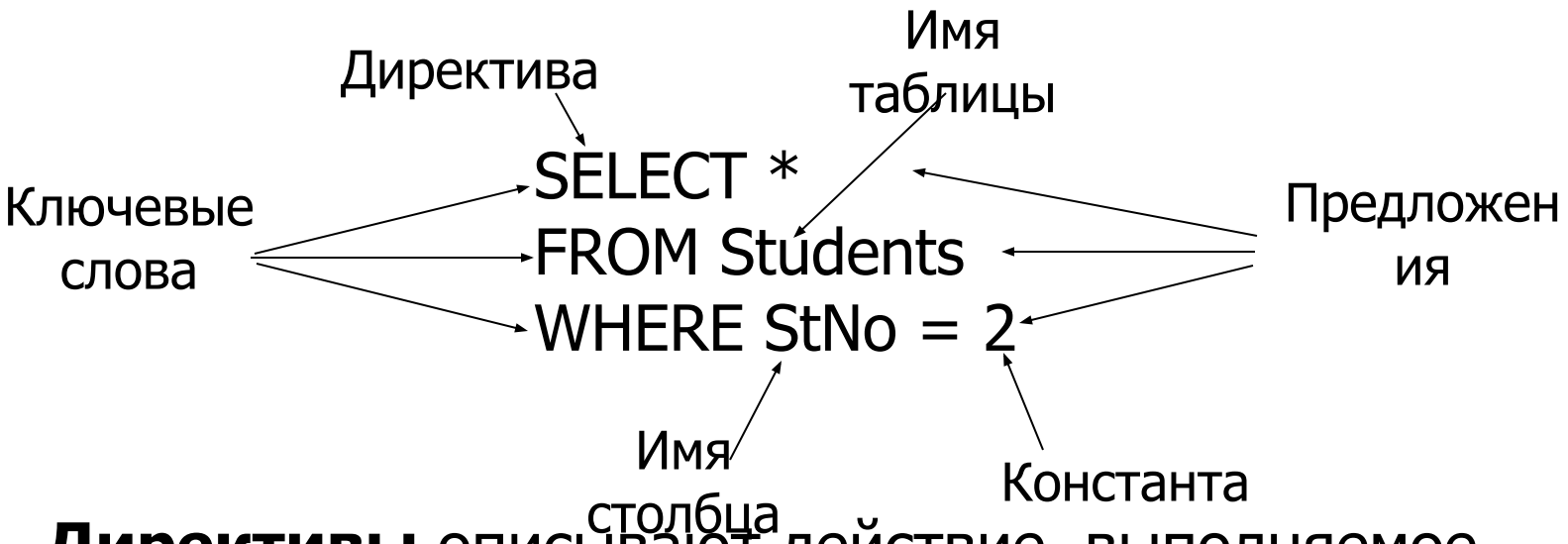
# Язык SQL

**SQL** (Structured Query Language – структурированный язык запросов) – язык для взаимодействия с БД.

# Основные понятия SQL

1. Структура **оператора** SQL




**Директивы** описывают действие, выполняемое оператором: SELECT (выбрать), CREATE (создать), INSERT (добавить), DELETE (удалить), UPDATE (обновить), DROP (удалить), ALTER (изменить), COMMIT (завершить и зафиксировать внесенные изменения), ROLLBACK (отменить внесенные изменения).

# Основные понятия SQL

**Предложение** описывает данные, с которыми работает оператор, или содержит уточняющую информацию о действии, выполняемом оператором: FROM (откуда), WHERE (где), GROUP BY (группировать по), HAVING (имеющий), ORDER BY (упорядочить по), INTO (куда).

2. **Имена (идентификаторы)**

- длина – до 128 символов
- используемые символы – только прописные или строчные буквы латинского алфавита, цифры или символ подчеркивания (_). *Первым символом* должна быть буква.
- составное имя – идентификатор базы данных, ее владельца и (или) объекта базы данных. Например, *полное имя таблицы* состоит из имени владельца таблицы и имени таблицы, разделенных точкой (.): Admin.Students.

# Основные понятия SQL

3. **Комментарии**
   - /* и */ – многострочный комментарий
   - -- – однострочный комментарий
4. **Типы данных**
   - INTEGER или INT – целое число (обычно до 10 значащих цифр и знак);
   - SMALLINT – "короткое целое" (обычно до 5 значащих цифр и знак);
   - NUMERIC(p, q) – десятичное число, имеющее p цифр (0<p<16) и знак; с помощью q задается число цифр справа от десятичной точки (q< p, если q = 0, оно может быть опущено);
   - REAL – число с плавающей запятой;

# Основные понятия SQL

- CHAR(n) (CHARACTER(n)) – символьная строка фиксированной длины из n символов (0 < n< 256);
- VARCHAR (n)  (CHARACTER VARYING (n))– символьная строка переменной длины, не превышающей n символов (n> 0 и разное в разных СУБД, но не более 8 Кб);
- DATE – дата в формате, определяемом специальной командой (по умолчанию mm/dd/yy, например, 10/03/12).
- TIME – время в формате, определяемом специальной командой, по умолчанию hh.mm.ss.
- BOOLEAN – принимает истинностные значения (TRUE или FALSE).

# Основные понятия SQL

## 5. Константы (литералы)

- Числовые константы: 21, -345, +234,6547
  - Константы с плавающей запятой: 1.5E3, -3.14159E1, 2.5 E7
- Строковые константы: ‘Это символьная строка’.

  Если в строковую константу нужно включить одинарную кавычку, то вместо нее надо писать две одинарные кавычки:

  ‘ Здесь внутри будут ‘ ‘ одинарные’ ’ кавычки’.
- Константы даты и времени. Пример для даты: ‘2012-10-03’, ‘1993-12-10’. Пример для времени: ’17:22:10’, ’01:01:01’.
- Логические константы: TRUE, FALSE, UNKNOWN
- Отсутствующие данные (значение NULL)

# Запросы на чтение данных. Оператор SELECT

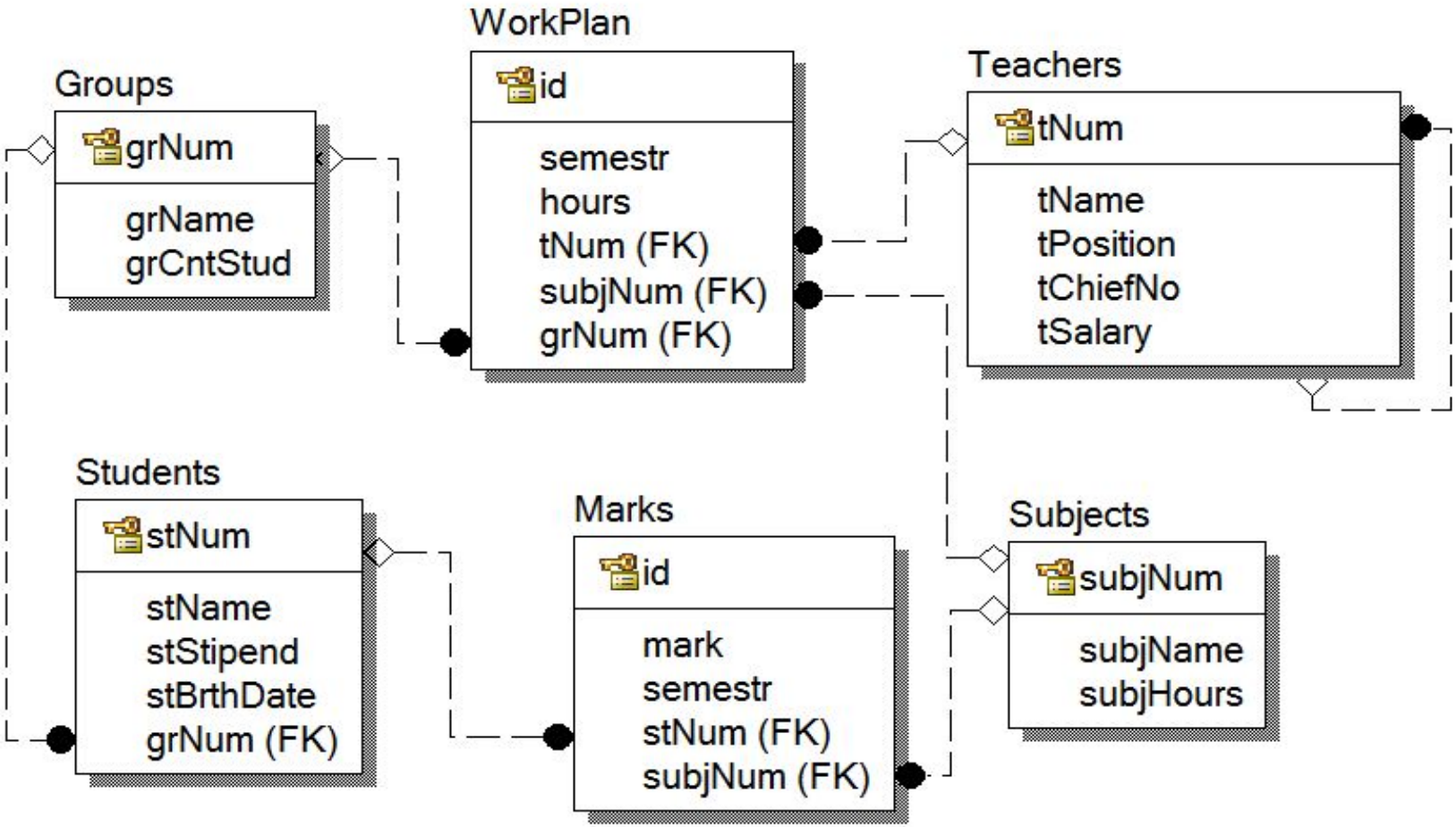
**Синтаксис оператора SELECT**

**SELECT** [**ALL** | **DISTINCT**] список_возвращаемых_столбцов|*
**FROM** список_имен_таблиц
[**WHERE** условие_поиска]
[**GROUP BY** список_имен_столбцов]
[**HAVING** условие_поиска]
[**ORDER BY** имя_столбца [**ASC** | **DESC**],…]

*Примечание*: в квадратных скобках указаны предложения, которые могут отсутствовать в операторе SELECT.

# Запросы на чтение данных. Оператор SELECT

**Схема БД (для примеров)**



# Запросы на чтение данных. Оператор SELECT

## 1. Цель запроса. Предложение SELECT

Предложение SELECT содержит список возвращаемых столбцов, разделенных символом «запятая» (,).

### 1.1 Способы указания выводимых столбцов

- **вывод значений определенных столбцов** одной из таблицы, указанной в предложении FROM

  Пример. Получить список студентов и размер их стипендий.

```
SELECT stName, stStipend
FROM Students;
```

# Запросы на чтение данных. Оператор SELECT

- для **вывода всех столбцов** таблицы, указанной в предложении FROM, можно перечислить все их названия или воспользоваться символом «звездочка» (*)

  Пример. Вывести все столбцы таблицы Groups.

```
SELECT grNum, grName, grCntStud
FROM Groups;
```

или

```
SELECT *
FROM Groups;
```

# Запросы на чтение данных. Оператор SELECT

- **уточнение имен** столбцов путем указания полного имени столбца: **имя_таблицы.имя_столбца**.

Пример:

```
SELECT Groups.grNum, Groups.grName, Groups.grCntStud
FROM Groups ;
```

# Запросы на чтение данных. Оператор SELECT

## 1.2 Исключение повторяющихся строк

Для исключения повторяющихся строк из результирующей таблицы используется ключевое слово **DISTINCT**, которое указывается перед списком возвращаемых столбцов.

Пример. Вывести значения столбца tPosition таблицы Teachers.

```
SELECT tPosition
FROM Teachers;
```

Пример. Вывести уникальные значения столбца tPosition таблицы Teachers.

```
SELECT DISTINCT tPosition
FROM Teachers;
```

# Запросы на чтение данных. Оператор SELECT

## 1.3 Использование вычисляемых выражений

Пример. Вывести фамилии студентов, размер их стипендий в грн. и в $.

```
SELECT stName, stStipend, stStipend / 8,14
FROM Students ;
```

# Запросы на чтение данных. Оператор SELECT

## 1.4 Переопределение имен результирующих столбцов

Для переопределения имени результирующего столбца (создания его синонима) используется ключевое слово **AS**.

# Запросы на чтение данных. Оператор SELECT

## 1.5 Включение текста в результат запроса

В предложении SELECT кроме имен столбцов и выражений с ними можно указывать константы (и константные выражения).

Пример. Вывести фамилии студентов и размер их стипендий, оформив результат предложениями на русском языке.

```
SELECT 'Студент', stName, 'получает стипендию', stStipend
FROM   Students ;
```

# Запросы на чтение данных. Оператор SELECT

## 2. Используемые таблицы. Предложение FROM

Предложение FROM содержит список имен таблиц, разделенных символом «запятая» (,).

Например, **FROM Students, Groups**.

Можно указывать **синонимы (псевдонимы) имен таблиц**.

Например, **FROM** Students **st**, Groups **gr**

# Запросы на чтение данных. Оператор SELECT

## 3. Отбор строк. Предложение WHERE

Предложение WHERE состоит из ключевого слова **WHERE**, за которым следует **условие поиска**, определяющее, какие именно строки требуется выбрать.

Если условие поиска имеет значение TRUE, строка будет включена в результат запроса.

Если условие поиска имеет значение FALSE или NULL, то строка исключается из результата запроса.

# Запросы на чтение данных. Оператор SELECT

## 3.1 Условия отбора строк

- **Сравнение**

  Выражение1 =|<>|<|>|<=|>= Выражение2

# Запросы на чтение данных. Оператор SELECT

- **Проверка на принадлежность диапазону значений (BETWEEN)**

проверяемое_выражение [**NOT**] **BETWEEN** минимум **AND** максимум

Пример. Получить список студентов, получающих стипендию в диапазоне от 650 до 1100 грн.

```
SELECT stName, stStipend
FROM   Students
WHERE stStipend BETWEEN 650 AND 1100;
```

# Запросы на чтение данных. Оператор SELECT

- **Проверка на принадлежность множеству (IN)**

проверяемое_выражение [**NOT**] **IN** (набор_констант)

Пример. Получить список студентов, получающих стипендию 650 или 730, или 900 грн.

```
SELECT stName, stStipend
FROM   Students
WHERE stStipend IN (650, 730, 900);
```

# Запросы на чтение данных. Оператор SELECT

- **Проверка на соответствие шаблону (LIKE)**

имя_столбца [**NOT**] **LIKE** шаблон [ESCAPE символ_пропуска), где

шаблон – это строка, в которую может входить один или более подстановочных знаков.

подстановочные знаки:

**%** – совпадает с любой последовательностью из нуля или более символов

Пример. Получить сведения о студентах, чья фамилия начинается с «Иван».

```
SELECT *
FROM   Students
WHERE stName LIKE 'Иван%';
```

# Запросы на чтение данных. Оператор SELECT

**_** (символ подчеркивания) – совпадает с любым отдельным символом.

Пример. Получить сведения о студентах, чье имя «Наталья» или «Наталия».

```
SELECT *
FROM   Students
WHERE stName LIKE '%Натал_я';
```

# Запросы на чтение данных. Оператор SELECT

символ пропуска используется для проверки наличия в строках символов, использующихся в качестве подстановочных знаков (%, _).

Пример. Получить сведения из таблицы "Data", где в поле результат содержится фрагмент текста "менее 50%" .

```
SELECT *
FROM Data
WHERE Result LIKE '%менее 50$% %' ESCAPE $;
```

# Запросы на чтение данных. Оператор SELECT

- **Проверка на равенство значению NULL (IS NULL)**

имя_столбца **IS** [**NOT**] **NULL**

Пример. Получить сведения о студентах, получающих стипендию.

```
SELECT stName, stNum, stStipend
FROM   Students
WHERE stStipend IS NOT NULL;
```

# Запросы на чтение данных. Оператор SELECT

- **Составные условия поиска (AND, OR и NOT)**

  WHERE [NOT] условие_поиска **[AND|OR]** [NOT] условие_поиска …

  Пример. Получить сведения о студентах, которые учатся в группе с кодом «1» и получают стипендию.

```
SELECT *
FROM   Students
WHERE (grNum = 1) AND (stStipend IS NOT NULL);
```

# Запросы с многими таблицами

**Естественное соединение таблиц**

Объединенную таблицу образуют пары тех строк из различных таблиц, у которых в связанных столбцах содержатся одинаковые значения.

Пример 1. Получить список студентов и названия их групп.

```
SELECT stName, grName
FROM   Students, Groups
WHERE (Students.grNum = Groups.grNum);
```

Связанные столбцы представляют собой пару «внешний ключ – первичный ключ».

# Агрегатные функции

**Агрегатная функция** принимает в качестве аргумента какой-либо столбец данных целиком, а возвращает одно значение, которое определенным образом подытоживает этот столбец.

- SUM(выражение | [DISTINCT] имя_столбца) – сумма [различных] числовых значений
- AVG(выражение | [DISTINCT] имя_столбца) – средняя величина [различных] числовых значений
- MIN(выражение | имя_столбца) – наименьшее среди всех значений

# Агрегатные функции

- MAX(выражение | имя_столбца) – наибольшее среди всех значений
- COUNT([DISTINCT] имя_столбца) – подсчитывает количество значений, содержащихся в столбце
- COUNT(*) – подсчитывает количество строк в таблице результатов запроса

**Примечание**: агрегатные функции нельзя применять в предложении WHERE

# Агрегатные функции

Пример 1. Найти суммарное, среднее, минимальное и максимальное значение стипендии студентов.

```
SELECT SUM(stStipend) AS Sm, AVG(stStipend) AS Av,
    MIN(stStipend) AS Mn, MAX(stStipend) AS Mx
FROM Students
```

Пример 2. Найти количество студентов, получающих стипендию.

```
SELECT COUNT(*) AS Cnt
FROM Students
WHERE stStipend > 0
```

# Сортировка результатов запроса. Предложение ORDER BY

ORDER BY имя_столбца [ASC | DESC], **...**

где, ASC – возрастающий, DESC – убывающий порядок сортировки.

Пример. Вывести список фамилий студентов, учащихся в группе КИ-125 в обратном алфавитном порядке.

**SELECT** stName

**FROM** Students, Groups

**WHERE** Students.stNum = Groups.grNum AND Groups.grName = ‘КИ-125’

**ORDER BY** stName DESC

# Запросы с группировкой. Предложение GROUP BY

Использование фразы GROUP BY позволяет сгруппировать строки в группы, имеющие одинаковые значения указанного поля:

| grName | | ORDER BY grName | | GROUP BY grName |
|---|---|---|---|---|
| КИ-121 | | КИ-101 | | КИ-101 |
| ПИ-111 | = | КИ-121 | = | КИ-121 |
| КИ-101 | | КИ-121 | | ПИ-111 |
| КИ-121 | | ПИ-111 | | |

К группам, полученным после применения GROUP BY, можно применить любую из стандартных агрегатных функций.

# Запросы с группировкой. Предложение GROUP BY

Пример 1. Получить список студентов и их средний балл.

```
SELECT stName, AVG(mark) AS AvgMark
FROM Students, Marks
WHERE Students.stNum = Marks.stNum
GROUP BY stName
```

**Примечание**. В списке отбираемых полей оператора SELECT, содержащего раздел GROUP BY, можно включать только агрегатные функции и поля, которые входят в условие группировки.

# Запросы с группировкой. Предложение GROUP BY

## Несколько столбцов группировки

Пример. Получить список студентов и их средний балл за каждый семестр.

```
SELECT stName, semestr, AVG(mark) AS AvgMark
FROM Students, Marks
WHERE Students.stNum = Marks.stNum
GROUP BY stName, semestr
```

## Значения NULL в столбцах группировки

Строки, имеющие значение NULL в одинаковых столбцах группировки и идентичные значения во всех остальных столбцах группировки, помещаются в одну группу.

# Запросы с группировкой. Предложение GROUP BY

**Условия поиска групп. Предложение HAVING**

Предложение HAVING, используемое совместно с GROUP BY, позволяет исключить из результата группы, неудовлетворяющие условию (так же, как WHERE позволяет исключить строки).

Пример 1. Получить список групп специальности КИ, в которых число студентов меньше 15.

```
SELECT grName, COUNT(*) AS CntStudents
FROM Students, Groups
WHERE Students.gtNum = Groups.grNum AND
Groups.grName LIKE 'КИ%'
GROUP BY grName
HAVING COUNT(*) <  15
```

# Вложенные запросы

**Вложенным запросом** (подзапросом) называется запрос, содержащийся в предложении WHERE или HAVING другого оператора SQL.

Пример 1. Получить список предметов, по которым была получена оценка <4.

```
SELECT subjName
FROM Subjects
WHERE subjNum IN ( SELECT subjNum
                   FROM Marks
                   WHERE mark < 4)
```

# Вложенные запросы

**Коррелируемым** подзапросом называется подзапрос, который содержит ссылку на столбцы таблицы внешнего запроса.

Пример 2. Вывести список студентов, средний балл которых выше 4,5.

```
SELECT stName
FROM Students
WHERE (SELECT AVG(mark)
        FROM Marks
        WHERE Marks.stNum = Students.stNum) > 4.5
```

# Вложенные запросы

**Особенности вложенных запросов**:

- вложенный запрос всегда заключается в круглые скобки;
- таблица результатов вложенного запроса всегда состоит из одного столбца;
- во вложенный запрос не может входить предложение ORDER BY.

# Предложение SELECT INTO

Для сохранения результатов SQL-запроса можно использовать новую таблицу. В этом случае синтаксис операции выборки имеет вид:

**SELECT** … **INTO** <имя новой таблицы>

**FROM** …

[**WHERE**…]

Пример: **SELECT** * **INTO** StudentsBackup

**FROM** Students

# Операции модификации данных (DML)

Для модификации данных используются три оператора: INSERT, DELETE и UPDATE.

1. Добавление строки в таблицу БД осуществляется с помощью оператора INSERT (вставка):

   **INSERT INTO** имя_таблицы (имя_столбца,…)

   **VALUES** (константа | NULL,…)

Пример. Добавить запись о новой группе ‘КИ-111’ в БД.

```
INSERT INTO Groups (grNum, grName, grHead)
VALUES (6, 'КИ-111', 11234);
```

# Операции модификации данных (DML)

При **добавлении значений во все столбцы** таблицы список столбцов можно не писать:

**INSERT INTO** Groups **VALUES** (6, 'КИ-111', 11234);

**Многострочный оператор INSERT** добавляет в таблицу несколько строк:

**INSERT INTO** имя_таблицы (имя_столбца,…) запрос

Пример. Скопировать данные из таблицы Groups в таблицу GroupsCopy

**INSERT INTO** GroupsCopy (grNum, grName, grHead)
**SELECT** grNum, grName, grHead **FROM** Groups;

# Операции модификации данных (DML)

2. Удаление строк из таблицы БД осуществляется с помощью оператора DELETE (удалить):

   **DELETE FROM** имя_таблицы

   [**WHERE** условие_поиска],

где условие поиска может быть вложенным запросом.

Пример. Удалить сведения о студенте с номером зачетки 12345.

```
DELETE FROM Students
WHERE stNum = 12345;
```

Примечание. Отсутствие предложения WHERE приводит к удалению ВСЕХ строк из указанной таблицы.

# Операции модификации данных (DML)

3. Обновление значения одного или нескольких столбцов в выбранных строках одной таблицы БД осуществляется с помощью оператора UPDATE (обновить):

   **UPDATE** имя_таблицы **SET** имя_столбца = выражение, ...

   [**WHERE** условие_поиска]

Пример. Увеличить на 20% размер стипендии студентов, которые ее получают.

**UPDATE** Students **SET** stStipend = 1.2 * stStipend

**WHERE** stStipend IS NOT NULL;

Примечание. Отсутствие предложения WHERE приводит к обновлению ВСЕХ строк из указанной таблицы.

# Операции определения данных (DDL)

**Команды DDL:**

- CREATE – создает объект БД;
- ALTER – изменяет определение существующего объекта;
- DROP – удаляет ранее созданный объект.

# Определение таблиц

**1. Создание таблиц с помощью языка SQL**

Для создания таблицы в языке SQL используется оператор CREATE TABLE:

**CREATE TABLE** <имя_таблицы> (

<имя_колонки> <тип_данных>[,

<имя_колонки> <тип_данных>]…)

[[**CONSTRAINT** <имя_ограничения>] <ограничение уровня колонки>]…

[[**CONSTRAINT** <имя_ограничения>] <ограничение уровня таблицы>]

# Определение таблиц

**Ограничения**:

**PRIMARY KEY** – определение первичного ключа таблицы;

**UNIQUE** – обеспечение уникальности значений в колонке;

**NULL / NOT NULL** – разрешение или запрещение неопределенных значений в колонке;

**CHECK <условие>** – задание условия на значение данных в колонке;

**[FOREIGN KEY <имя_колонки>] REFERENCES <имя_таблицы> <имя_колонки>** – определение внешнего ключа для таблицы.

# Определение таблиц

**Пример**:

```
CREATE TABLE student (
numZach integer CONSTRAINT pkSt PRIMARY KEY,
fio char(30),
stipend integer CHECK (stipend BETWEEN 500 AND 800),
pol char(1) CHECK (pol='м' OR pol='ж'),
grNum integer REFERENCES groups (grNum) ON DELETE
      CASCADE
);
```

# Определение таблиц

**2. Изменение таблиц. Оператор ALTER TABLE**

**ALTER TABLE** имя_таблицы
**ADD** определение_столбца
**ALTER** имя_столбца
**SET DEFAULT** значение | **DROP DEFAULT**
**DROP** имя_столбца **CASCADE** | **RESTRICT**
**ADD** определение_первичного_ключа
**ADD** определение_внешнего_ключа
**ADD** условие_уникальности_данных
**ADD** условие_проверки
**DROP CONSTRAINT** имя_ограничения
**CASCADE** | **RESTRICT**

# Определение таблиц

Пример. Добавить первичный ключ в таблицу student

**ALTER TABLE** student **ADD CONSTRAINT** “pk”
**PRIMARY KEY** (numZach);

# Определение таблиц

## 3. Удаление таблицы

Для удаления таблицы из БД в языке SQL используется оператор DROP TABLE:

**DROP TABLE** <имя_таблицы> **CASCADE** | **RESTRICT**